JPL Control/Structure Interaction Test Bed Real-Time Control Computer Architecture

Presented at the

3rd Annual Conference on

Aerospace Computational Control

Oxnard, CA

28-30 August, 1989

By

Dr. Hugh C. Briggs, Deputy Manager

JPL CSI Program

Jet Propulsion Laboratory

California Institute of Technology

Pasadena, CA

Control/Structure Interaction Test Bed Real-Time Control Computer Architecture

Hugh C. Briggs, Deputy Technical Manager
Jet Propulsion Laboratory
4800 Oak Grove Drive
Pasadena, CA 91109

Abstract

The Control/Structure Interaction Program is a technology development program for spacecraft that exhibit interactions between the control system and structural dynamics. The program objectives include development and verification of new design concepts - such as active structure - and new tools - such as a combined structure and control optimization algorithm - and their verification in ground and possibly flight test. A focus mission spacecraft has been designed based upon a space interferometer and will be the basis for design of the ground test article. The ground test bed objectives include verification of the spacecraft design concepts, the active structure elements and certain design tools such as the new combined structures and controls optimization tool. In anticipation of CSI technology flight experiments, the test bed control electronics must emulate the computation capacity and control architectures of space qualifiable systems as well as the command and control networks that will be used to connect investigators with the flight experiment hardware.

The Test Bed facility electronics have been functionally partitioned into three units: a laboratory data acquisition system for structural parameter identification and performance verification; an experiment supervisory computer to oversee the experiment, monitor the environmental parameters and perform data logging; and a multilevel real-time control computing system. The design of the Test Bed electronics is presented along with hardware and software component descriptions. The system should break new ground in experimental control electronics and will be of interest to anyone working in the verification of control concepts for large space structures.

The NASA Control/Structure Interaction Program

The NASA CSI Program is an element of the Control of Flexible Structures Task in the NASA Civilian Space Technology Initiative. Three NASA Centers participate in the CSI Program: Langley Research Center, Marshall Space Flight Center and the Jet Propulsion Laboratory. This multiyear program to develop and validate new design technologies is organized around five elements: Systems and Concepts, Analysis and Design, Ground Test Methods, Flight Experiments and Guest Investigation Program. The CSI program goal is to develop validated technology that will be needed to design, verify and operate interactive control/structure systems to meet the ultraquiet structure requirements of 21st century NASA missions.

The CSI Program will integrate the advances made in other discipline technology programs to make the new spacecraft design methodology (see Figure 1). Controls programs such as Computational Control will develop a new generation of tools for multibody simulation, multibody component representation, and control analysis and synthesis. Structures technology programs such as Computational Mechanics are developing advanced finite element analysis codes. CSI will integrate these tools into a multidisciplinary environment and develop additional tools such as simultaneous structure and control optimization methods and conceptual design tools for flexible spacecraft structure/control architectures. New CSI systems and concepts, such as active structure, will be developed and integrated into the focus mission design example.

Other developments that will enable high-performance, flexible spacecraft design include an investigation of microdynamics and development of ground test methods for controlled flexible spacecraft structures. Microdynamic characterizations of spacecraft components such as joints and struts will identify the linearity of typical elements when dynamic motions are restricted to the submicron regimes required for future spacecraft. In addition, disturbance sources will be characterized at the microdynamic level to support analysis of ultraquiet spacecraft systems.

Implementation of the CSI Methodology - The Design Environment

The design environment is an important element of the methodology and consists of several elements. The following section will address the computer systems and the laboratory testing facilities. The software and analytical tools are described in the companion paper on the design methodology. [1]

The CSI computer system is a distributed network-based system consisting of workstations and servers (Figure 2). Laboratory testing computers are attached to the network to support the close integration of verification tests to the development of systems concepts and tools. Sufficient commercial technology exists to support a heterogeneous equipment set based upon standard network interfaces. For example, systems from Apple, DEC, Sun, Apollo, HP and others can all participate in an Ethernet network using TCP/IP. (For a brief description of terms, see the glossary.) This capability supports various user preferences and capabilities and provides the mechanism to protect existing corporate investments in computer systems.

The distributed system utilizes servers for those functions not allocated to the per-engineer workstations. Large computers, such as a CRAY or departmental VAX, function as compute servers to provide an execution site for large, computationally intensive jobs. Other servers might provide specialized capabilities for animation, data base management or communications. Most workstation companies make it financially attractive to collect most of the system disk resources in one or more file servers that support some form of a network disk system (eg. Sun's NFS). These file servers are repositories for large data sets, system executables and application libraries.

The workstations must support the interactive design environment with excellent speed and graphics. The CSI methodology requires computation of intermediate sized (ie. 100+ states)

---

1. "Control/Structure Interaction Design Methodology," H.C. Briggs and W.E. Layman, Proceedings 3rd Annual Conference on Aerospace Computational Control, Oxnard, CA, 28-31 August, 1989.

problems and presentation of solid models on the workstations. The derived requirements for workstations are: 3-10 MIP 32 bit CPU, 12-16 Mb memory, Unix operating system, 200 Mb disk, Ethernet interface, 3-D vector graphic accelerator and windowed presentation manager with a mouse.

The network environment also extends into the laboratory where verification and validation experiments are executed on the CSI Test Bed. The computing environment internal to the lab is shown in Figure 3. The four functions are: real-time control, experiment supervision, modal analysis and software development. Individual systems can be readily purchased to perform each function although it is possible to configure certain commercial systems to perform multiple duties. In any case, the software development system will probably not be instantiated in the laboratory, using individual analyst workstations and the experiment supervisory computer instead.

The real-time control computer system will be described in more detail in the following sections but a summary is presented here to support the environment description. The control computer will be a distributed, multiprocessor computer based upon commercial VMEbus products. The operating system supports remote consoles, software loading, a prioritized scheduler and shared memory message passing. An excellent example is VxWorks from Wind Rivers although the underlying kernel requires additional multiprocessor extensions. Analysts will prepare simple control subroutines on their workstation and produce a load module just as they would any program for execution. Remote login facilities are provided for access to any real-time CPU and a C-like shell provides the operator interface. Products such as DbxWorks provide source level symbolic debugging.

The experiment supervisory computer provides the laboratory operator console and overall control of the Test Bed. This system monitors and logs environmental variables such as temperature and air velocity, monitors a panic button during experiment execution and collects measurements from the external truth sensor. Remote access from any network workstation allows remote execution of experiments.

The modal analysis and data acquisition system is a standard commercial product and supplies a necessary function found in all dynamics laboratories. To characterize the structural dynamics of the test article, a modal survey can be performed utilizing a large number of accelerometers distributed over the structure. This is typically done to verify open loop system models but should also be an integral part of closed loop system performance measurement. Results are available to any analyst via the network.

The laboratory environmental requirements are quite severe. Noise and seismic disturbance constraints will require all personnel and actively cooled electronics to be in an adjacent control room. During tests, the test chamber must be unoccupied, closed, and carefully maintained at constant temperature. This will require development of control procedures for remote experiments and forms the basis for emulation of on-orbit flight experiments. The essential Shuttle command, communication, and control features can be readily emulated with the network-based computer system and the computational capabilities of space-qualified computers can be replicated in the ground test hardware. Figure 4 illustrates the scale and complexity of a test bed that models a space-based interferometer.

CSI Testing Requirements

CSI will validate the system concepts, components and tools in realistic ground tests. Where the ground environment precludes acceptable verification due to such effects as the gravity field, seismic and acoustic disturbances and size limitations, flight tests will be proposed to complete the development and validation of the technology.

Testing is recognized as an essential component of the design process. The design methodology will include close coupling of the analysis with the testing and evaluation of results. This will foster verification of new system concepts and designs as well as provide analytical support for new ground test techniques. In addition, the CSI flight experiments will be designed to develop techniques for extending ground testing methods to on-orbit flight tests.

As a result of integrating testing into the design process, several capabilities must be built into the ground test facility. Interactive evaluation of control system performance must be provided to explore system phenomena and to enable reconciliation of measured behavior with predicted behavior. To validate the new optimization methods and to evaluate system robustness properties, substitution of any structural element will be provided without dismantling large subsections of the test article. Support for remote investigation of system performance via the electronic network, already mentioned as a requirement for CSI analysts, will also include support for off-site Guest Investigators. This access includes all test measurement data as well as the control programs of the real-time control computer. Finally, emulation of all essential Shuttle command, communication, and control features that impact proposed flight experiments will be provided.

RTC Requirements

Given the CSI program goals and testing requirements, several requirements for the real-time controller are presented in the following. Most are functional requirements that have shaped the system architecture although the section will be closed with a general statement of the computational requirements.

The real-time controller and the Test Bed are an integral part of the CSI design environment. As such they are part of a distributed network system that must support remote access and remote software development. This has been stated for the experiment supervisor and the data acquisition system but also applies to individual control CPUs. The Test Bed control electronics will execute selected portions of the FMI control functions including path length control, wavefront streering and active structure control. In addition, the system may be called upon to generate certain disturbance profiles for actuators. Because the system must serve as a validation host for spacecraft system designs, analysis tools, and methods, it must have a reconfigurable topology, flexible software, and a range of speed capabilities. For example, in addition to supporting ground test with the best possible execution speed, the system must emulate the restricted computational capabilities expected in the flight experiment environment.

The software development environment for the real-time control system must be particularly simple. Transparent cross-compilation and mainstream languages such as C and

Fortran are required. Since the system connectivity will be quite complex, remote symbolic source level debugging tools will be required. Libraries are required for I/O interface routines and kernel access functions. In summary, software development for the verification of new control techniques must be well supported to minimize costs.

The computational requirements are readily stated in terms of control loop parameters as follows. Although converting these to speed and size requirements for system components is not straightforward, a certain amount of experience with similar systems exists and a prototype of the controller is proposed to verify system performance issues. The Test Bed control system consists of many loosely coupled control loops. These typically utilize less than 10 states and 5 sensors and actuators. The fastest loop operates at a maximum of 600 Hz. The exception to the typical loop is the active structure controller that drives the active struts. This loop can control up to 20 struts, each having one actuator and two sensors. This requirement is made significantly more demanding by the possibility of loop rates to 1000 Hz although only one instance of an active structure controller will exist at any time. Single precision (32 bit) floating point arithmetic is acceptible and analog interfaces operate with 16 bit integers.

System Functional Architecture

Figure 5 shows the functional units of the laboratory in the top level structure chart. Compare this figure with Figure 3, Test Bed Computing Environment. The functions are shown as boxes and the real-time controllers are shown as the object labelled 2.0. Signal paths are shown as directional arrows and the signal content is indicated with text labels. At this high level, the controllers and the test article are simply shown exchanging the data "Controls - Actuators & Sensors."

This figure also shows the network connections between the analysts (represented by the stub "World Access") and the supervisory computer, real-time controllers and the modal analysis & data acquisition system. The analysts access each system as remote consoles using the remote login services of the network. The experiment supervisory computer and the data acquisition system each have local operator consoles.

The second level structure charts provide further definition of unit functions. For example, Figure 6 shows more details of the real-time controllers and contains objects labeled 2.x to show the heirarchy. The six functions in Figure 6 are taken from the FMI Control System Functional Diagram reproduced as Figure 7. Most of the information exchanged between functions in the FMI diagram represent mechanical or optical mechanisms and have been deleted in the structure chart to leave only information to be transmitted by the real-time control system.

Each of the six functions shown in Figure 6 has been further defined in lower level structure charts and one of these is shown in Figure 8. The active struts used in the active structure control might have local controllers which can command the strut motor and read the strut sensors. Each strut loop might be designed to present an idealized actuator to the block labeled " 2.31 Active Structure Control" which provides the strut commands.

Implementation Concept

The functional structure charts break the real-time control into progressively smaller pieces which can be assigned to hardware units for execution. Prior to this, however, an implementation concept must be selected. This will consist of choices for compute elements, busses, interconnects and software.

The CSI control system will be purchased as commercial items to the maximum extent possible. This policy should lead to reasonable costs, short development time and minimum work force requirements. Economical products with sufficient capacity exist at the board level, although custom integration of the desired components does not seem to be offered at economical prices. This also applies to the system software where operating systems and device interfaces are commercially available.

The implementation (see Figure 9) will provide separate communication channels for operator access and real-time data. Each CPU will be logically attached to the network with Ethernet connections to each chassis. This channel will be used for operator login, software loads, status display and debugging. Within a chassis, the network will be carried over the back plane with one CPU designated as the gateway.

The real-time data will be passed over the back plane as shared memory messaging. This leads to a need to locate CPUs with large message volumes in the same chassis and to extend chassis with bus bridges where chassis capacities are exceeded. Bus bridges that utilize the multi-master capabilities of the back plane are required to avoid degrading overall system performance.

The software implementation will utilize a commercial real-time kernel and development environment such as VxWorks. The kernel requires multiprocessor extensions and support for a wide variety of CPUs. VRTX with MPV meets these needs. Analog I/O interface routines will be based upon vendor supplied libraries and typically require only simple services such as single sampling of sequential channels. A standard control module will be written based upon a linear, constant coefficient update law and supplied to analysts. This template can be sized as required and filled with data to implement most common control laws.

RTC Typical Crate

This implementation concept leads to a standard chassis to host the functions contained in the structure charts. Approximately ten chassis might be required to house all of the units shown in Figure 9. Each chassis - or "crate" - will be configured similarly but execute a particular control function and be connected to different sensors and actuators.

The typical chassis is shown in Figure 10 and utilizes a 21 slot VMEbus rack mounted cabinet. Each chassis will have a CPU to allocate crate resources and handle network communications. This CPU will drive the chassis Ethernet controller and serve as a gateway, providing TCP/IP access to other chassis CPUs over the back plane. Chassis resources might consist of one or more array processors and 4 to 16 Mbytes of memory.

Each chassis might contain 3 to 5 additional CPUs to host control functions. Analog I/O devices are provided for each CPU to achieve maximum system speed and avoid contentions. These CPUs will initially utilize fast MC68030 processors with MC68882 floating point units and fast local memory. Future upgrades will augment this with DSP or RISC based single board computers supported by the development system and the real-time kernel.

The software on each SBC consists of the VRTX kernel with MPV, the VxWorks shell, device drivers, and the control loop code. The kernel supports task scheduling and message passing while the shell handles the operator interface and the symbol table. SBCs with at least 1 Mbyte of memory provide sufficient storage to meet these needs.

Software Implementation

Software development for complex real-time systems such as the CSI Test Bed can pose many problems. This system is clearly multirate and consists of many loosely coupled control loops. Without care, the system software can prove to be unmanageable.

The objectives of the software implementation design are all oriented toward simplifying the resulting system. The modularity of the structure charts will be followed by coding individual functions. This will promote structure, independence and simplicity in the resulting software system. The Test Bed is a research and development vehicle and must support evolving and untried algorithms. To meet throughput requirements, the software must execute in multiprocessor hardware and achieve maximum possible speed consistent with a simple, high level language software development environment. Finally, selected sections of the software must be hosted in the future on advanced DSP or RISC processor hardware.

The selection of a common real-time kernel with an operator shell and development environment supports these objectives. Standard procedures for kernel access, message passing and resource allocation are provided. Beyond this, analog I/O routines and other locally written standard modules will be maintained in libraries. Initially, a simple control law will be written and supplied as a template. The coefficients and size of this law can be changed but the module interface can be standardized.

Control Law Examples

The constant coefficient, linear update control law is illustrated in Figure 11. The module consists of four separate functions for input, state estimation, control generation and output. The modules pass prearranged data messages via the kernel messaging services. Each module is seen by the kernel scheduler as a process that, after initialization, is sleeping while waiting for the arrival of the messages. Each function executes sequentially but asynchronously with the exception of the input which waits for the next time slice to begin. If synchronous output is required, the final module could also be scheduled on time slice boundaries.

The software for each module in this control law can be readily standardized. The device interface library can store drivers that have been parameterized by base address and number of channels. The estimater and control routines require the actual coefficients of individual control laws but are based upon the standard template. With this structure, future changes should be localized to a few modules.

For certain control architectures, this software can be readily extended to a multiple processor hardware system. An example based upon decentralized control is shown in Figure 12. The estimator and control routines have been replicated and instantiated with the specific coefficients of two controllers. The message passing services of the kernel are used to pass the real-time data and control the execution sequencing of the routines. In this case, the message addresses include the CPU number but are otherwise unchanged from the single processor example. The execution sequence is again initiated by the input routine which is scheduled for the next time slice.

The capability to easily prepare multiprocessor (and multirate) systems is based upon the multimaster features of the VMEbus hardware and the multiprocessor services of the real-time kernel. Complex systems can be written, checked out in a modular fashion on a single CPU and expanded to multiple CPU systems after the logic and data structures have been verified. With care, loosely coupled control loops should realize near-linear increase in speed with the addition of CPUs.

Closing Remarks

A system cost model has been built based upon a spread sheet program. Hardware is allocated to crates at the board level and chassis capacity and total costs are calculated. A separate spread sheet contains a library of board components with note annotations documenting vendor, configuration, and discounts. This cost model was used to support quantity, capability and scope trade studies in a very effective, interactive manner.

A system prototype will be constructed prior to a critical design review in March 1990. This prototype will be used to verify performance and implementation issues such as CPU capacity, interconnect speeds and the complexity of the operator interface. The real-time control system must be fabricated prior to the initial turn-on of the Test Bed in October 1991 for system check-out.

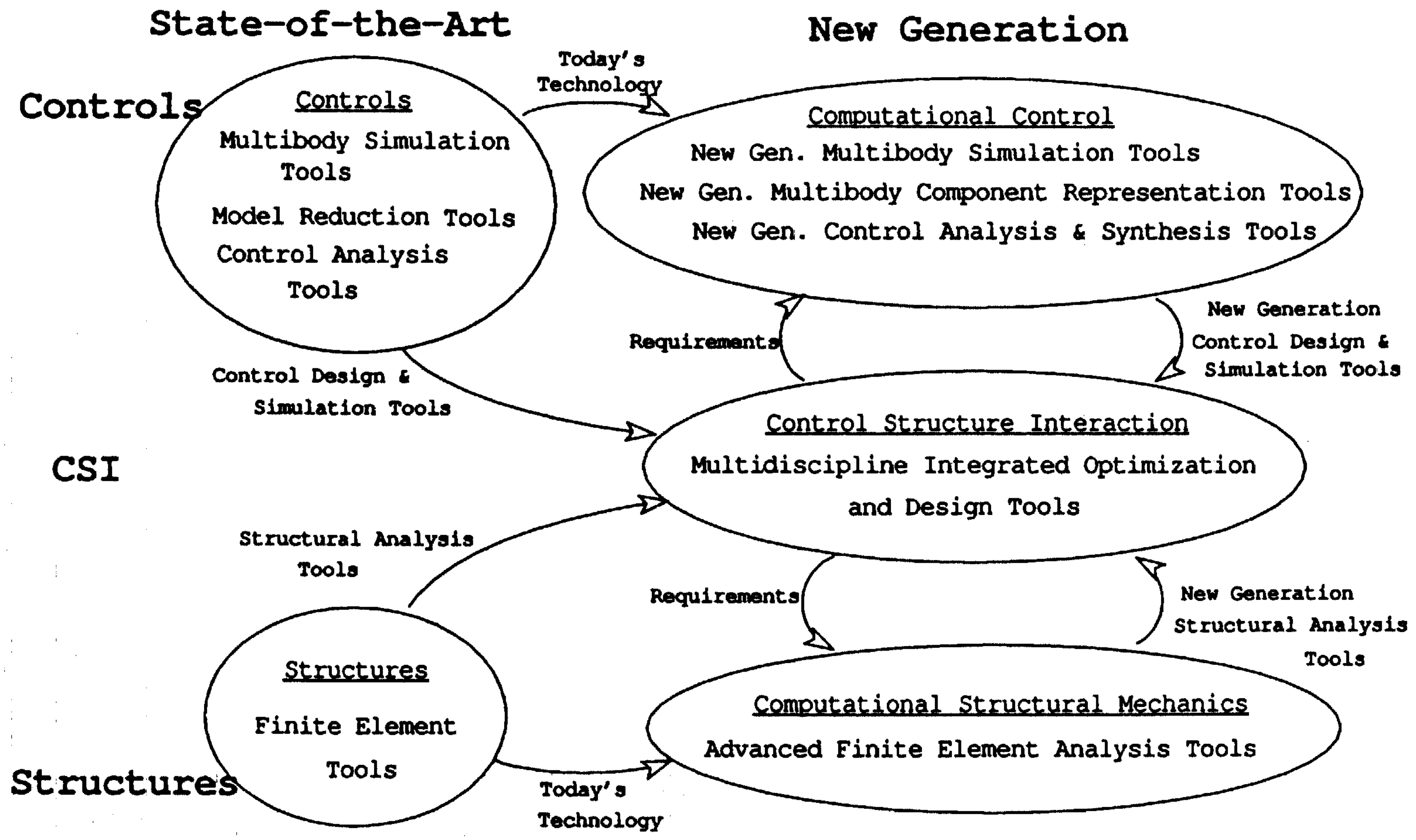
Acknowledgement

The research described in this paper was performed as part of the Control Structure Interaction (CSI) program at the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration.

Glossary

CPU - Central Processing Unit, the core of a computer.
DSP - Digital Signal Processor, a small fast CPU typically for embedded control applications.
FMI - Focus Mission Interferometer, a CSI design example.
MPV - Multiple processor extensions to the VRTX kernel.
RISC - Reduced Instruction Set Computer - new fast CPU.
SBC - Single Board Computer.
TCP/IP - Ethernet protocol for local area networks.
VRTX - Real-time kernel from Ready Systems.
VxWorks - Real-time operating system from Wind Rivers.

Figure 1. Relationship Between Controls, Structures, and CSI Tools


State-of-the-Art
New Generation
Controls
CSI
Structures
Controls
Multibody Simulation Tools
Model Reduction Tools
Control Analysis Tools
Today's Technology
Computational Control
New Gen. Multibody Simulation Tools
New Gen. Multibody Component Representation Tools
New Gen. Control Analysis & Synthesis Tools
Requirements
New Generation Control Design & Simulation Tools
Control Design & Simulation Tools
Control Structure Interaction
Multidiscipline Integrated Optimization and Design Tools
Structural Analysis Tools
Requirements
New Generation Structural Analysis Tools
Structures
Finite Element Tools
Computational Structural Mechanics
Advanced Finite Element Analysis Tools
Today's Technology

ORIGINAL PAGE IS
OF POOR QUALITY

Figure 2. CSI Computing Network

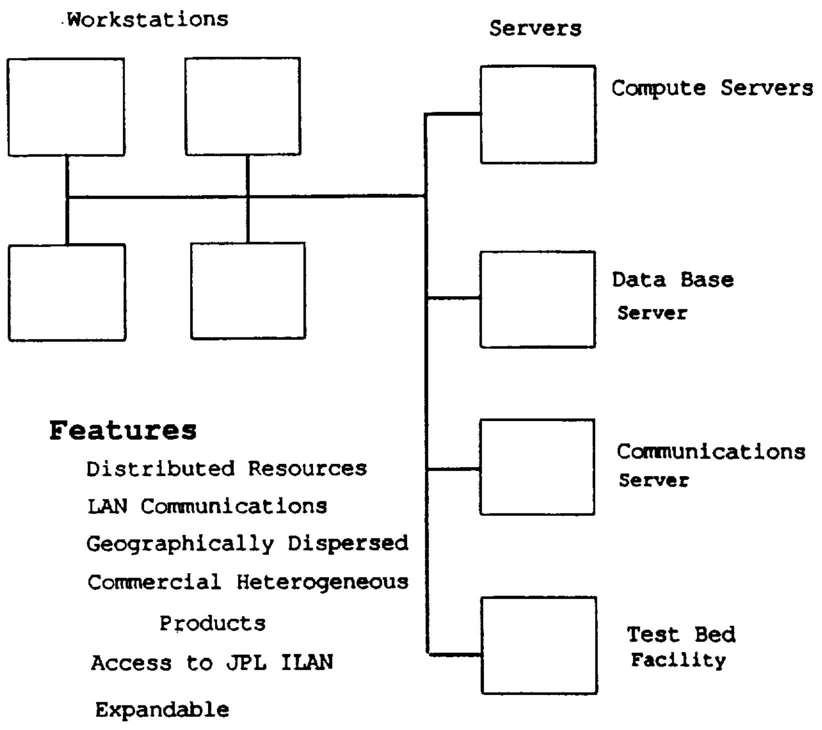
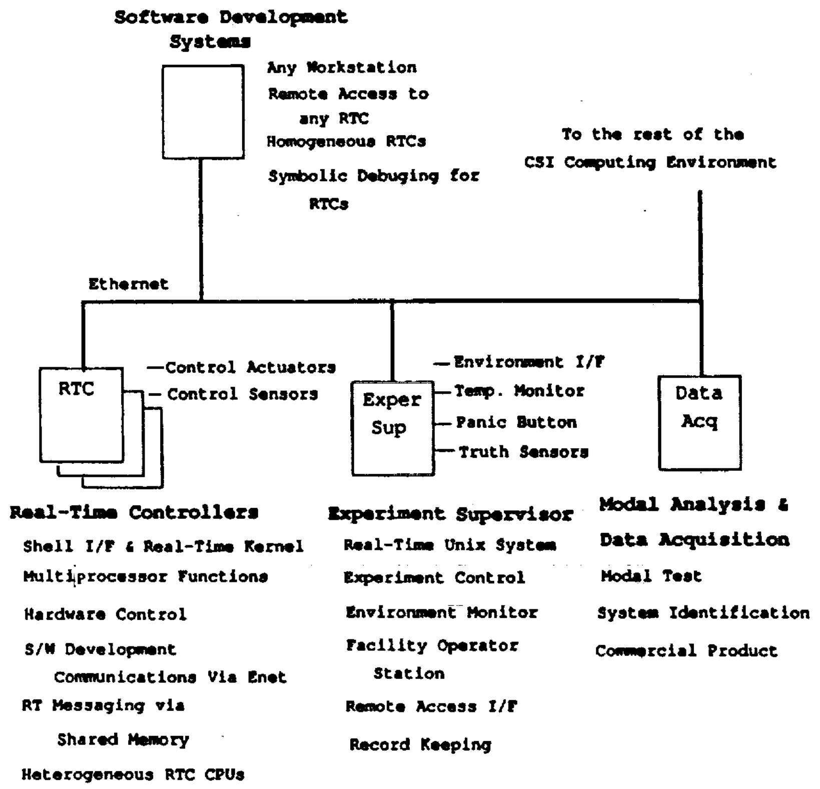


Figure 3. Test Bed Computing Environment



JPL

Figure 4. Integrated Controls and Structures Laboratory

CARL

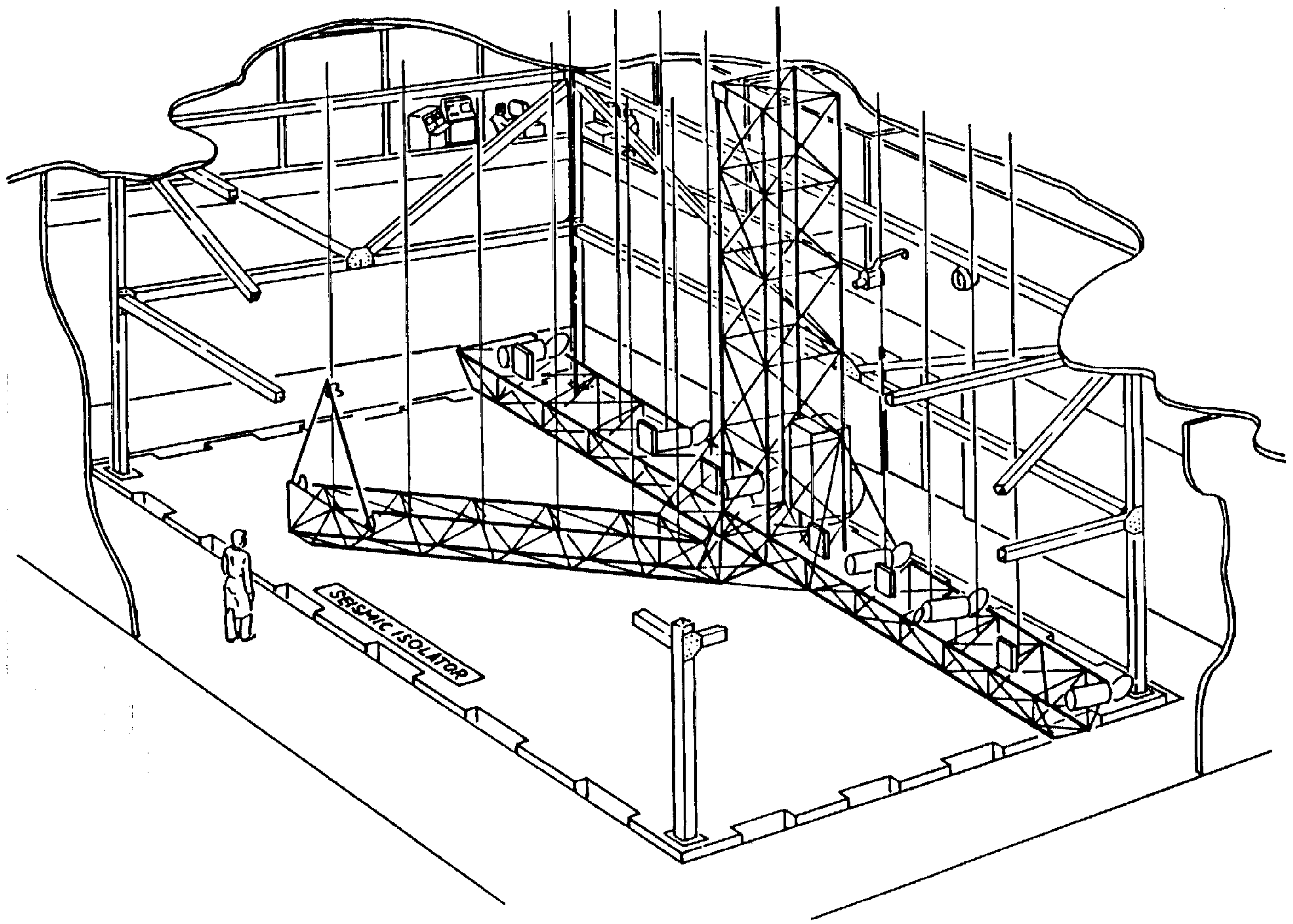


Figure 5. RTC Laboratory Environment

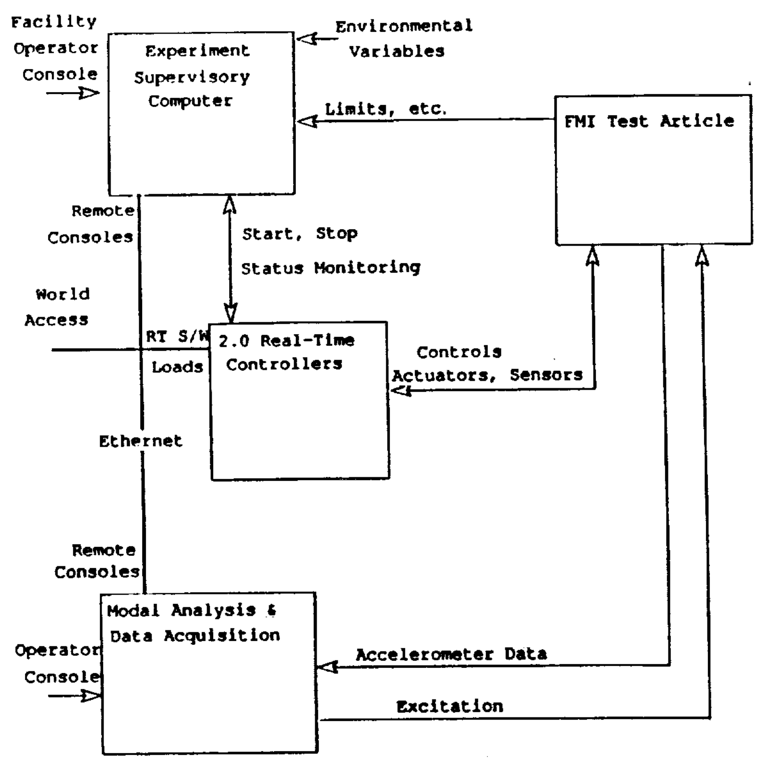
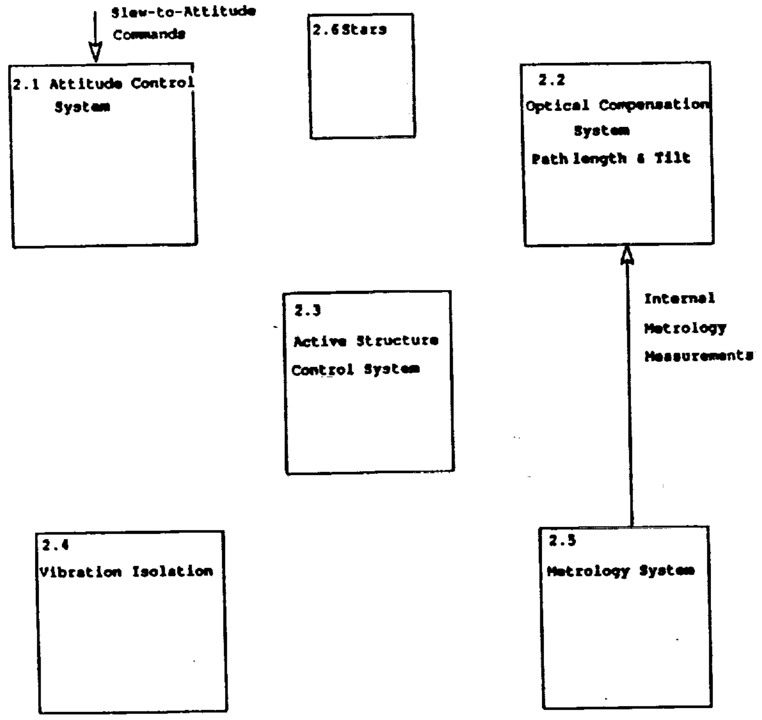


Figure 6. 2.0 Real-Time Controller*

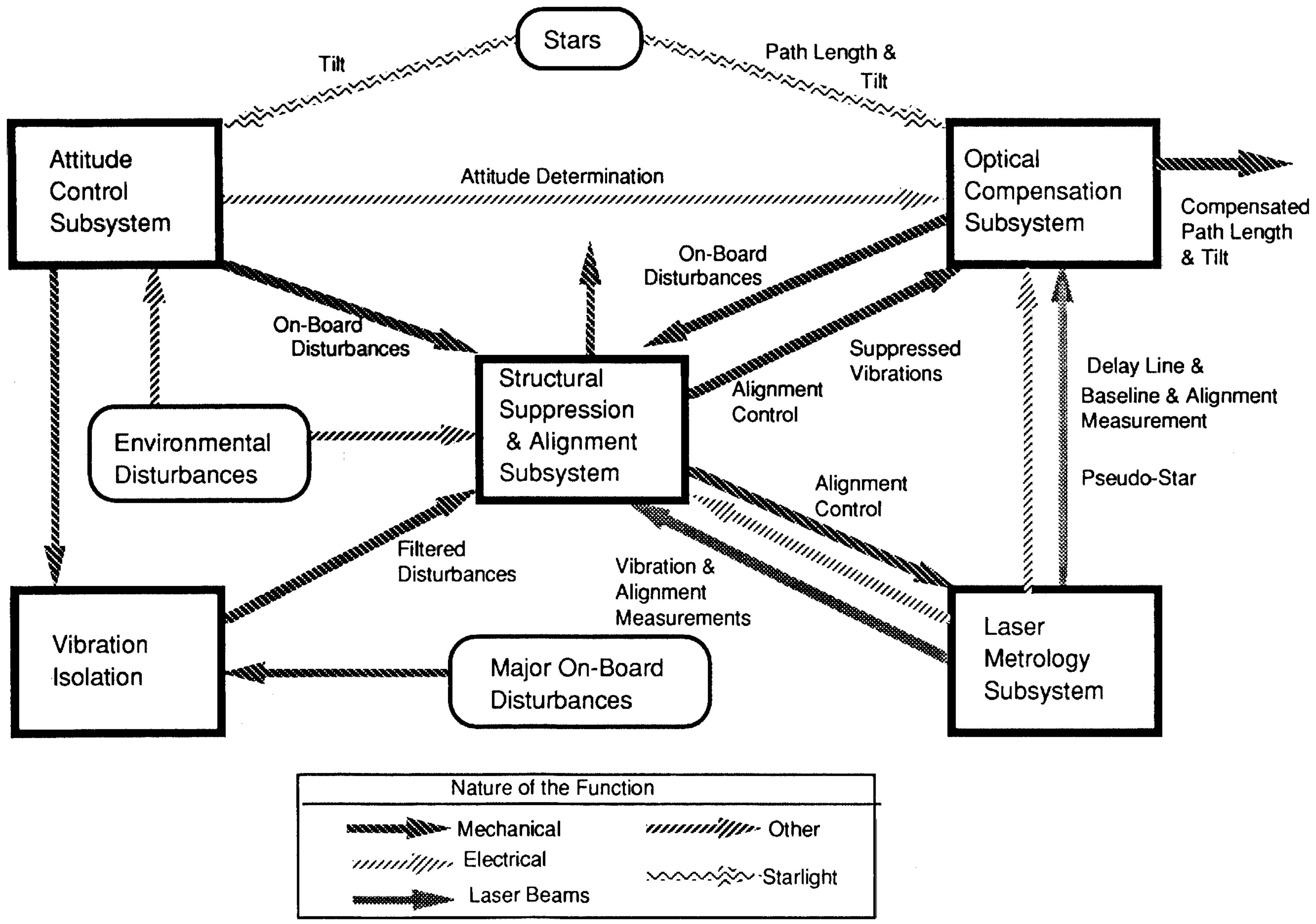


Slew-to-Attitude Commands
2.1 Attitude Control System
2.6 Stars
2.2 Optical Compensation System Path length & Tilt
2.3 Active Structure Control System
Internal Metrology Measurements
2.4 Vibration Isolation
2.5 Metrology System

*Based on the FMI Control System Functional Diagram

JPL

Figure 7. Focus Mission Interferometer Top Level Functional Block Diagram



Stars
Tilt
Path Length & Tilt
Attitude Control Subsystem
Attitude Determination
Optical Compensation Subsystem
Compensated Path Length & Tilt
On-Board Disturbances
On-Board Disturbances
Structural Suppression & Alignment Subsystem
Alignment Control
Suppressed Vibrations
Delay Line & Baseline & Alignment Measurement
Pseudo-Star
Environmental Disturbances
Alignment Control
Filtered Disturbances
Vibration & Alignment Measurements
Vibration Isolation
Major On-Board Disturbances
Laser Metrology Subsystem
Nature of the Function
Mechanical
Electrical
Laser Beams
Other
Starlight

Figure 8. 2.3 Active Structure Control System

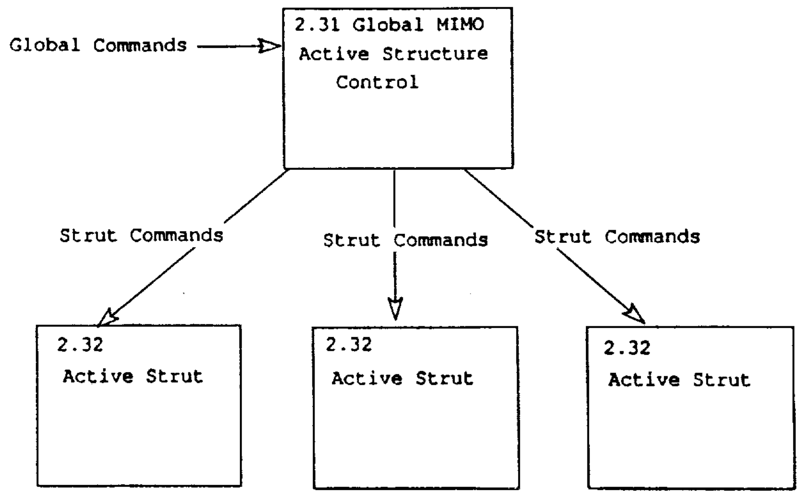


Figure 10. RTC Typical Crate

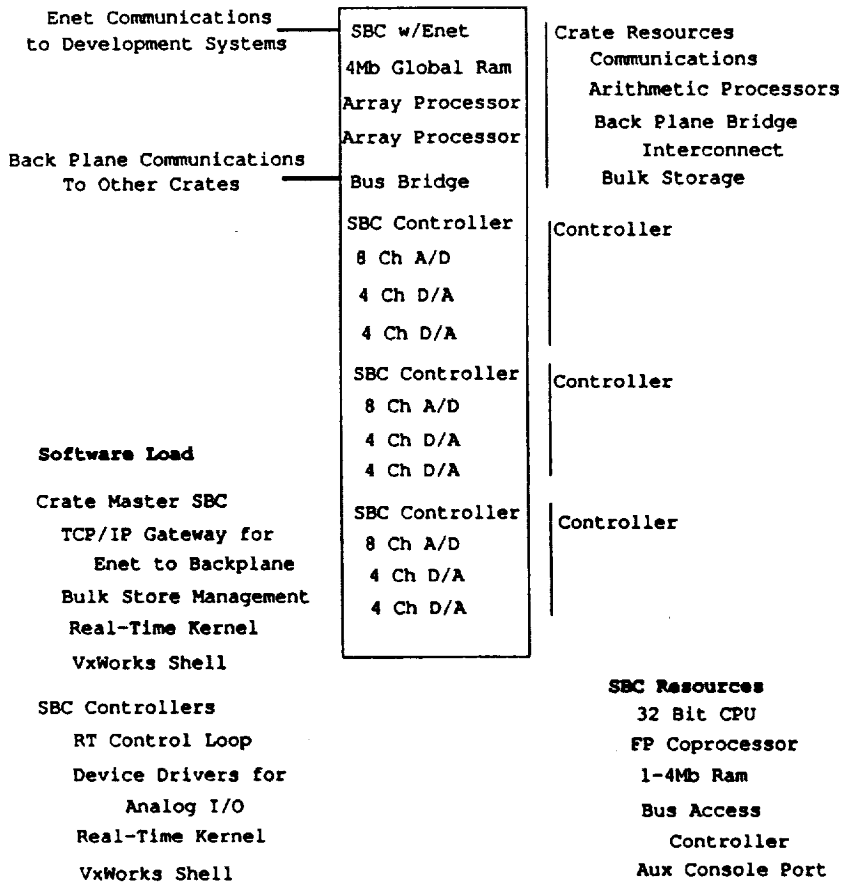


Figure 9. RTC Implementation Concept

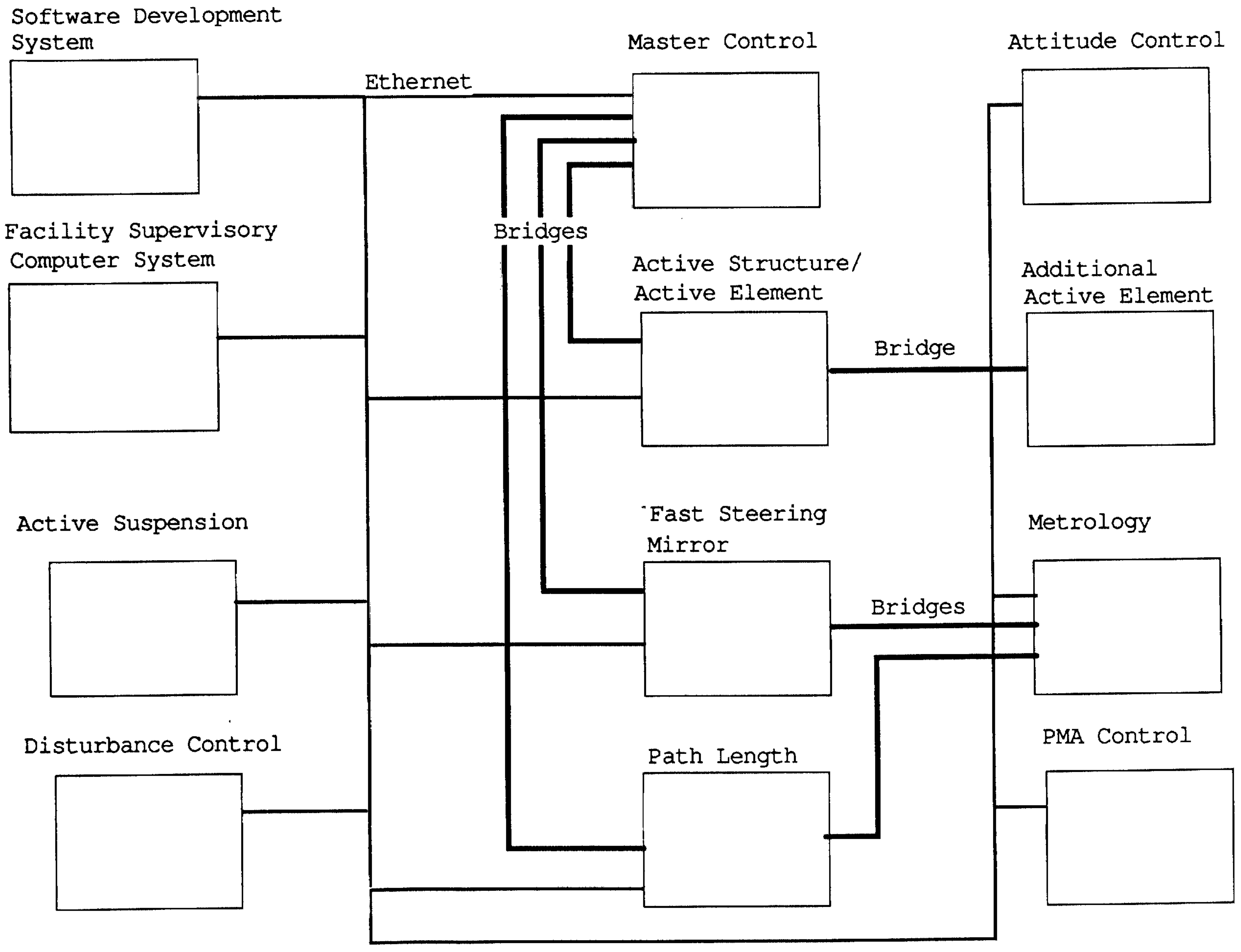


Figure 11. RTC Control Software
Functional Structure

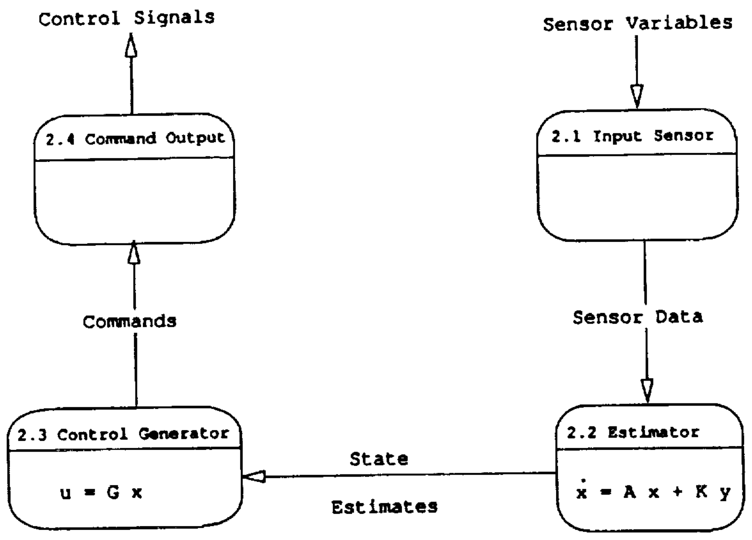
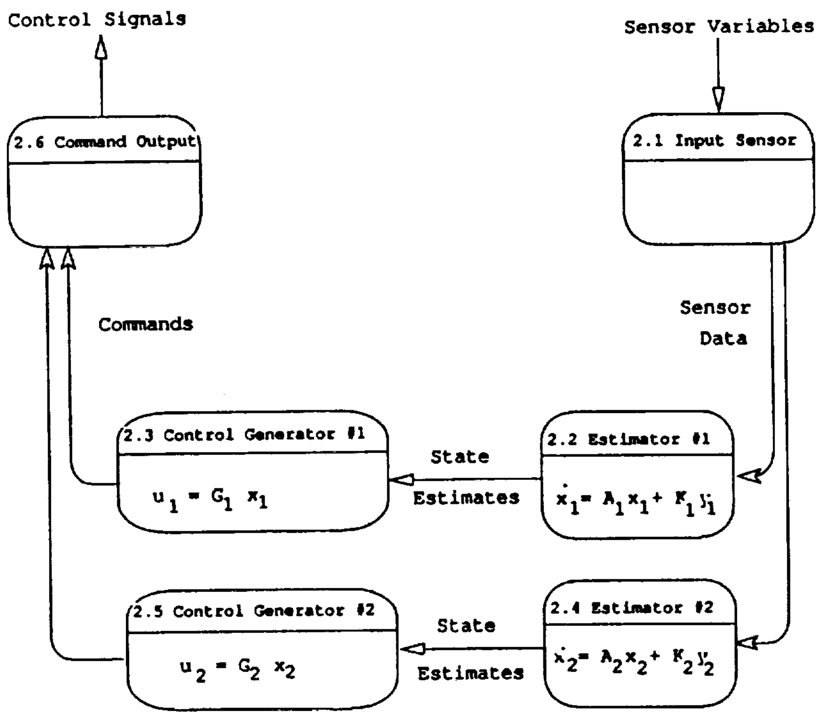


Figure 12. RTC Control Software
Decentralized Control Example



Control Signals
Sensor Variables
2.6 Command Output
2.1 Input Sensor
Commands
Sensor Data
2.3 Control Generator #1
$u_1 = G_1 x_1$
State
Estimates
2.2 Estimator #1
$\dot{x}_1 = A_1 x_1 + K_1 y_1$
2.5 Control Generator #2
$u_2 = G_2 x_2$
State
Estimates
2.4 Estimator #2
$\dot{x}_2 = A_2 x_2 + K_2 y_2$